PBCdlComm

Version 1.3.6

User Manual

2009/07/16

Table of Contents

Introduction

The open-source PBCdlComm software is designed for data acquisition from PakBus-protocol based Campbell Data loggers by the Scientific Data Management team at Pacific Northwest National Laboratory (www.pnl.gov). This effort was funded by US Department of Energy's Atmospheric Radiation Measurement program (www.arm.gov). As an alternative for Campbell Scientific's Loggernet software, this is ideal for deploying with rugged field PCs that have limited processing resources and do not require a full-blown GUI module. The software is designed so that it can be extended to implement a new data storage mechanism, a customized post-processing module and/or integrated with web-services for sophisticated monitoring or reporting.

Organization of this document

- "System requirements and Installation" describes how to compile the application.
- "Usage" describes the process of configuring the application for deploying in field.
- "Data format" briefly presents the output data format.
- "Application Development" explains the program architecture, read only if you are interested in modifying/extending the code base.

System requirements and Installation

The software uses the following libraries, C++ STL, libxml2 and log4cpp. It is being used on Red Hat Linux based production systems since 2006.

To compile the application, edit the Makefile to set -I and -L switches appropriately as illustrated below. *No edits are required* if the libraries are installed at a standard location as /usr/include (header files) and /usr/lib (shared libs).

```
IFLAGS   += -I/home/choudhury/apps/install/Linux-i686/include
LFLAGS   += -L/home/choudhury/apps/install/Linux-i686/lib -llog4cpp -lpthread
```

Edit the BIN_NAME and OUT_DIR entries in the Makefile to change the output binary name and the output directory.

Usage

The software is designed to be run from a command line interface. Typically the execution command would be built dynamically using a Perl or Shell script and be run from a scheduler application as crontab.

Command Line Options

To display the command line options, type “pbcdl_comm –h” at the command line.

Usage :
pbcdl_comm [-c config_file_path] [-d] [-w working_path] [-p connection string] [-h] [-v]

Options :

- -c Specify the complete path of the collection configuration file. The format of the configuration file is described in the remainder of this document.
- -d Set log level to debug. This will also enable packet level communication logging. A log file with “Comm” prefix will be opened in the directory specified as “Working Path”.
- -p Specify the connection string. Currently only serial communication is supported. Using port name /dev/ttyS0 and 115200 baud rate needs to be specified in a comma-delimited string with no spaces as “/dev/ttyS0,115200”.
- -w Working path refers to the directory where all data and log files will be stored. Specifying this from command line overrides the working path mentioned in configuration file. Apart from writing data files and log files in the “WorkingPath” directory, the software also creates a <WorkingPath>/.working directory, where all the temporary data files are stored and application’s runtime state is persisted.
- -h Print this help message
- -v Print version information

Example Configuration File

The application requires a configuration file to be specified through the “-c” command line switch. Optionally, a number of entries in the configuration file can be overridden by other command line switches.

The configuration file was designed with extensibility in mind; XML was chosen due to it’s flexibility in describing hierarchical information.

For deploying to multiple sites/remote locations, one might choose to describe the common settings through a configuration file which would be released to all sites and set some of the site-specific options using the command line switches through a Perl/Shell wrapper.

Specifics:

The vtime parameter is intended as a measure of the latency of the link between the datalogger and the host. The time of a serial port read can be approximated as (0.1*vtime) seconds. The software uses a vtime value of 10 if no other value is specified in a configuration file. One may want to raise the value of vtime if there are too many signature errors reported in the logfile. For a direct connection between the host and the datalogger, a vtime value as low as 2 can be used.

The sample_int_secs and the file_span_secs parameters are useful when the data is collected in ASCII data files. The sample_int_secs is the sampling interval for a specific table and file_span_secs suggest the frequency at which one wishes to create a new data file.

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- CONFIG FILE TO SETUP DATA COLLECTION FROM PAKBUS LOGGER -->
<!-- Comments with "**" will need to be set for deployment  -->
<!-- ** Set the station name to a brief description. -->
<COLLECTION logger="" station_name="">
    <!—Serial Port configuration -->
    <CONNECTION type="serial">
        <!-- ** Set the port_name to the serial port address -->
        <port_name>/dev/ttySI0</port_name>
        <!-- ** Set the baud rate, if different -->
        <baud_rate>115200</baud_rate>
        <vtime>10</vtime>
    </CONNECTION>
    <!--  Turn on debugging : use TRUE/FALSE -->
    <DEBUG>TRUE</DEBUG>
    <!—Data output options -->
    <DATA>
        <WORKING_PATH>
        /data/collection/rld/rldmfrsrS1.00/collecting/.pbcdl_comm
        </WORKING_PATH>
       <!--
         Add a table entry to collect data from it. The
         file_span_secs parameter can be used to set file
         durations. sample_int_secs parameter indicates the
         update frequency, which is used to determine if a data
         file is complete and be moved up from the
         .../.working directory.
        -->
        <COLLECT_TABLE>
            <TABLE sample_int_secs="20"file_span_secs="3600">
                Table155
            </TABLE>
            <TABLE sample_int_secs="86400" file_span_secs="3600">
                TableHdr
            </TABLE>
        </COLLECT_TABLE>
    </DATA>
    <!—PakBus protocol settings  -->
    <!-- The following are default values -->
    <PAKBUS>
      <DST_PAKBUS_ID>1</DST_PAKBUS_ID>
      <DST_NODE_PAKBUS_ID>1</DST_NODE_PAKBUS_ID>
      <!-- Default value for security code is zero -->
     <SECURITY_CODE>0</SECURITY_CODE>
    </PAKBUS>
</COLLECTION>
```

Data Format

The currently available implementation supports generating hourly ASCII data files in the "TOA5" format supported by Campbell Scientific. Please refer to the Loggernet manual (www.campbellsci.com/documents/**manuals**/**loggernet**.pdf) for a detailed description of the format. The content of a sample data file is shown below. The number inside "[]" on the left indicate line numbers.

```
[1] "TOA5","E04MFRSR","CR1000","2118","CR1000.Std.11","CPU:SUNAE5J6BR_E4.CR1","33826","Table155"
[2]"TIMESTAMP","RECORD","xoset","az","el","al","cosz","cosasz","pamass","vBat","Thead","THTC","Tlogger","Dnadi
   r(1)","Dnadir(2)","Dnadir(3)","Dnadir(4)","Dnadir(5)","Dnadir(6)","Dnadir(7)","Deast(1)","Deast(2)","Deast(3)","Deast
   (4)","Deast(5)","Deast(6)","Deast(7)","Dsun(1)","Dsun(2)","Dsun(3)","Dsun(4)","Dsun(5)","Dsun(6)","Dsun(7)","Dwe
   st(1)","Dwest(2)","Dwest(3)","Dwest(4)","Dwest(5)","Dwest(6)","Dwest(7)","kount","khome"
[3]"TS","RN","sec","deg","deg","deg","""","Volt","degC","degC","degC","mV","mV","mV","mV","mV","mV","mV","mV","m
   V","mV","mV","mV","mV","mV","mV","mV","mV","mV","mV","mV","mV","mV","mV","mV","mV","mV","mV","mV","--
   ","--"
[4]"","","Smp","Smp","Smp","Smp","Smp","Smp","Smp","Smp","Smp","Smp","Smp","Smp","Smp","Smp","Smp","Smp"
   ,"Smp","Smp","Smp","Smp","Smp","Smp","Smp","Smp","Smp","Smp","Smp","Smp","Smp","Smp","Smp","Smp","S
   mp","Smp","Smp","Smp","Smp","Smp","Smp","Smp","Smp"
[5]"2008-05-01
15:00:20.0",245844,0,79.26,38.4,38.42,0.621,0.621,1.607,11.98699,40.1,39.92,26.78,92.8,89,94.2,93.5,97.7,89.5,8
8.8,91,87.2,92.1,91.4,95.6,87.5,87.1,25.14,35.45,28.91,24.73,24.87,23.05,20.69,90.2,86.4,91.3,90.6,94.7,86.8,86.1,
282,498
```

Note: Unlike Loggernet, the seconds component of the data record is a floating point number. This is added in view of the microsecond component of the NSec time structure in the PakBus protocol.

Architecture

The application was originally intended for supporting PakBus protocol based devices. The latest version present a generic API for developing data collection processes for diverse instruments/data loggers, where each data source would be supported as a separate module.

One of the prime objectives of the API is to design for extensibility. The current version does not implement all possible options for data sources or persistence mechanisms; however, the goal is to provide the right interfaces so that an implementation of a new data persistence mechanism or a different communications interface can be easily integrated into the software.

Secondly, the design attempts to minimize the coupling between the following aspects of the software:

Process Abstraction

The data collection process is modeled through a *DataCollectionProcess* interface, where each implementation will be build on different combinations of the following aspects.

Data Protocol

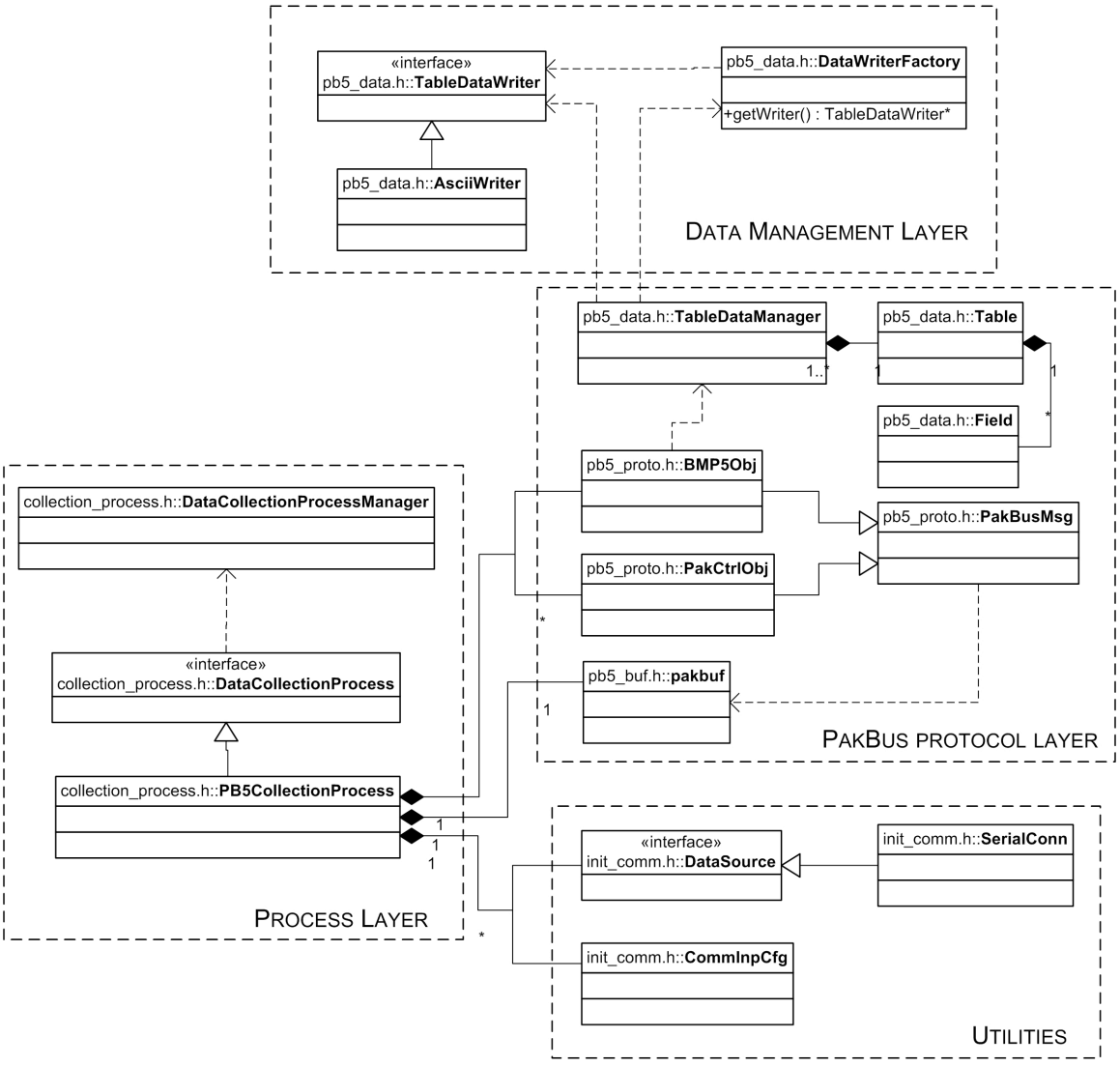
The current implementation supports the PakBus protocol developed by Campbell Scientific. Please refer to www.**campbell**sci.com/documents/manuals/bmp5-**transparent-commands**.pdf for the protocol details. The data protocol might evolve in future, or new protocols might emerge. In that case, a new implementation of the *DataCollectionProcess* will encapsulate the new/updated protocol details.

Data Management

A common way for collecting data is storing it in ASCII data files. However, many applications prefer to use a database for storing the records. It is not uncommon to find applications that may prefer a different storage format. A common requirement of sensor data collection applications is monitoring data in real-time. The *TableDataWriter* interface is designed to provide a base for implementing all such logic. It provides a series of callback functions that would be invoked at different points of the processing cycle while parsing the binary data downloaded from the logger. By implementing the callback functions, an application can store the data in a database, perform post-processing or even invoke a web-service for notification when a specific event is observed.

Data Source

The PakBus protocol is a communications-interface agnostic data protocol. The current version of the PbCdlComm software only supports serial communications interface. Support for TCP/IP communications might be added in the future. The goal of the



DataSource interface is to hide the communication specific issues from the process and treat different communication interfaces in a transparent fashion. When multiple communications are supported, the user must be able to configure the application at run-time by specifying the "connection string" using the command-line interface or outlining the connection details in the configuration file.

The above diagram presents the primary design components that implement the above mentioned aspects. *It does not show all the components or functions present in the source. Please refer to the Doxygen generated documentation for complete details.*

Contact

Sutanay Choudhury
Pacific Northwest National Laboratory
Email: sutanay.choudhury@pnl.gov
Phone: +1-509-375-3978